Genetically Evolved Transformations for Rescaling Online Handwritten Characters

Deepu V. and Sriganesh Madhvanath

Abstract— **In this paper, we describe using genetically evolved rescaling transformations for scale normalization of online handwritten characters. GP is used for evolving the rescaling functions such that the transformed characters make less recognition errors. The procedure for evolving these transformations requires only labeled characters for training and testing the classifier, but no knowledge of the internals of the classifier. It therefore holds promise as a general mechanism for improving the recognition performance of any online HWR algorithm.**

I. Introduction

Handwriting input to a computing device facilitate a natural and easy interface for text input. A position sensing device tracks writing and outputs a stream of x-y coordinates. Segmentation of input writing into basis symbols of the language is carried out explicitly (asking user to write in boxes, finding spatial and/or temporal gaps in the input) or implicitly (combined with recognition). In either case the core algorithm is a shape recognizer that accepts a segmented chunk of x-y coordinates and labels them as a particular basic symbol [1]. A shape recognizer typically consists of a preprocessor (to remove any possible variations) and a classifier. In online handwriting recognition two common variations that can happen is (a)difference in scale and (b)difference in velocity. A common strategy for compensating scale difference is to rescale the character to a unit square. This approach ignores the aspect ratio information and renormalizes every character to the same size.

In this work, we try to investigate the use of Genetic Programming (GP) to evolve a transformation for a better preprocessing algorithm. GP is an attempt at getting computers to learn to solve problems without being explicitly programmed [2], [3]. GP differs from the other learning techniques in that it seeks solutions directly in the form of computer programs.

In Section II we present an overview of genetic programming. Section III deals with formulation of the problem and selection of parameters. Experimental results are presented in Section IV. Section V concludes the paper and provide some future vectors.

II. Genetic Programming

Large class of problems in artificial intelligence and machine learning require discovery of computer programs that produce some desired output for particular inputs. The process of solving such problems can be reformulated as a search for a highly fit individual computer program in the space of possible computer programs. In particular the search space is the space of all possible computer programs composed of functions and terminals appropriate to the problem domain. Genetic Programming (GP) provides a way to search for this fittest individual computer program. Analogous to genetic algorithms the search starts with an initial random population and uses the genetic method to hit upon the best fit solution. Genetic methods differ from most other search techniques in that they simultaneously involve a parallel search involving hundreds or thousands of points in the search space.

A. Representation

Individual structure that undergo adaptation in genetic programming are hierarchically structured computer programs. Size shape and contents of these computer programs can dynamically change during the process.

The set of possible structures in GP are the set of all possible compositions of functions that can be composed recursively from the set of N_f functions from $F = \{f_1 f_2, \ldots, f_{N_f}\}$ and the set of N_t terminals from $T = \{a_1, a_1, \ldots, a_{N_t}\}$. Each function f_i takes $z(f_i)$ arguments. The functions can include arithmetic operations, mathematical functions, logical and relational operators, recursive and iterative functions etc. Terminals are typically constants (numbers, boolean "true" etc), variables (inputs, sensors or state variables of a system), or functions that take no arguments. The search space of genetic programming is the space of all possible functions which can be recursively created by compositions of available functions and terminals for the problem. The solution space can equivalently be viewed as a space of rooted point labeled trees with ordered branches having internal points labeled with available functions and external points (leaves) labeled with available terminals.

In genetic programming terminal set and function set should be selected to satisfy two requirements.

closureeach function in the function set should be well defined and closed for any combination of arguments it may encounter. This ensures arbitrarily created functions from compositions of available functions and terminals result in valid expressions.

sufficiencyires set of terminals and primitive functions capable of expressing a solution to the problem.

B. Initial Creation of Population

Initial generative process can be implemented in several different ways resulting in initial random trees of different

The authors are with the Hewlett-Packard Laboratories, Bangalore 560 030, India. E-mail: deepuv@hp.com, srig@hp.com.

0-7803-8909-3/04/$20.00 ©2004 IEEE

sizes and shapes. Two of the basic ways are called the "full" and the "grow" method.

The *"full"* method of generating the initial random population involves creating trees for which the length of every nonbacktrackingpath between an endpoint and the root is equal to maximum depth specified.

The *"grow"* method of generating initial random population involves growing trees that are variably shaped. The length of a path between an endpoint and the root is no greater than specified maximum depth.

The *"ramped half and half"* generative method is a mixed method that incorporates both the full method and grow method. It involves creating an equal number of trees using a depth parameter that ranges between 2 and maximum specified depth.

C. Fitness Measure

In nature the fitness of an individual is the probability that it survives to the age of reproduction and reproduces. In the Artificial world fitness is measured in some way and this measurement is used to control the application of the operations that modify the structures in the artificial population. The most common approach to measuring fitness is to create an explicit fitness measure for each individual in the population.

D. Operations for Evolution

The operations generally employed in GP for evolution are selection, crossover, and mutation. Sometimes other methods like permutation, editing, encapsulation, editing etc., are also used.

1) **Selection** Selection is the basic engine for Darwinian natural selection and survival of the fittest. Selection consists of two steps – 1. an individual is selected from the population according to some selection method based on fitness, 2. selected individual is copied without alternation from current population to new population. Different algorithms for selection includes
   - *fitness proportionate selection* individual is copied to next generation will be proportional to the fitness of that individual
   - *rank selection* depends only on the rank of individual fitness, not on the fitness values.
   - *tournament selection* group of individuals are chosen at random from the population and one with best fitness is selected.
2) **Crossover** Crossover operation for genetic programming creates variation in population by producing new offspring from parts taken out of each parent. In general parents are chosen by the same method used for selection operation.
3) **Mutation** Mutation introduces random changes in population structures. Mutation is beneficial in reintroducing diversity in a population that might otherwise tend to converge prematurely.

E. Termination Criteria

The termination criterion commonly employed is to stop either after a pre-specified maximum number of generations are over or after some problem specific *success predicate* has been satisfied. The success predicate often involves finding a 100% correct solution to the problem.

III. Problem Definition

For the purpose of investigation we assume that the input symbols are already segmented and re-sampled (for velocity compensation) to a constant number of points.

The problem may be stated as follows.

Given:

- An N-class character classifier that reads normalized symbols which can be trained and evaluated.
- A set of M labeled symbols with representative distribution over the pattern classes, and
- No knowledge of the internals of the classifier,

Objective:

To discover a transformation for "scale normalization" that improves recognition of the classifier in a (near-) optimal fashion (Figure 1).

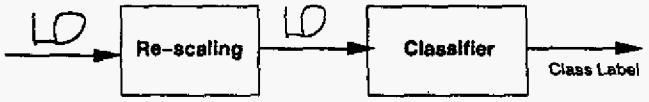


Fig. 1. Block Diagram

In order to use GP, we first define (i) functions and terminals, (ii) fitness measure and (iii) operational parameters. Subsequent sections discuss each of these in turn.

A. Functions and Terminals

The input symbol is a sequence of coordinates $C = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, our aim is to find scale factors K_{x_i} and K_{y_i}, such that the output of the rescaling module will be

$$x_i' = \frac{x_i}{K_{x_i}} \tag{1}$$

$$y_i' = \frac{y_i}{K_{y_i}} \tag{2}$$

K_x and K_y are assumed to be arbitrary functions of the following factors, which generally describes the scale information about a symbol.

$$K_{x_i} = f_x(Sx, Sxx, maxX, minX, x_i) \tag{3}$$

$$K_{y_i} = f_y(Sy, Syy, maxY, minY, y_i) \tag{4}$$

where,

$$
\begin{aligned}
Sx &= \sum_{i=1}^{n} x_i \\
Sxx &= \sum_{i=1}^{n} x_i^2 \\
maxX &= arg\ max\ _i\{x_i\} \\
minX &= arg\ min\ _i\{x_i\} \\
Sy &= \sum_{i=1}^{n} y_i \\
Syy &= \sum_{i=1}^{n} y_i^2 \\
maxY &= arg\ max\ _i\{y_i\} \\
minY &= arg\ min\ _i\{y_i\}
\end{aligned}
$$

We restrict f_x and f_y to be of the same form, i.e, the scaling algorithm in both directions are same. The terminal set contains the elements $\{S, S2, Max, Min, Val\}$ where $\{S = Sx$ or Sy , $S2 = Sxx$ or Syy, $Max = maxX$ or $maxY$ $Min = minX$ or $minY$ and $Val = x_i$ or y_i. Also some constants $K1$ and $K2$ are included in the terminal sets.

The function set we choose includes arithmetic operators $(+, -, *, /)$, relational operators $(<, >)$ and conditional operator $(if - then - else)$.

B. Choice of Fitness Measure

The fitness of a transformation can be characterized in many ways. We have chosen to do so directly in terms of classifier performance in the rescaled character set. We divided the input character set into training and test sets. Both training and test sets are rescaled using the GP output. The classifier is trained using the training set. Recognition accuracy is evaluated using the test set and used as the fitness measure.

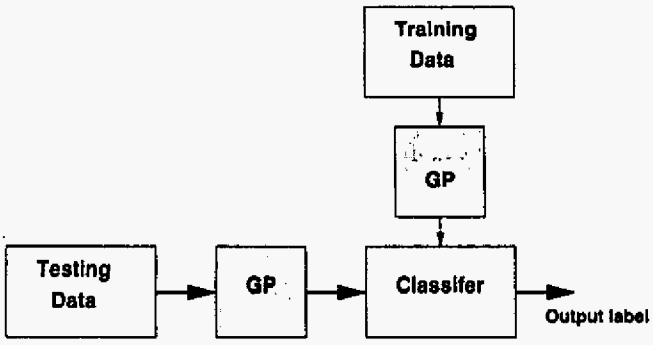


Fig. 2. Fitness measure evaluation

C. GP Parameters

We use a standard set of parameters for GP runs, as tabulated in Table I.

IV. Experimental Evaluation

We use an isolated handwritten character database in Tamil. The database consists of 161 isolated symbols that occur in the Tamil language. There are 15 writers, each one contributed

TABLE I
GP PARAMETERS

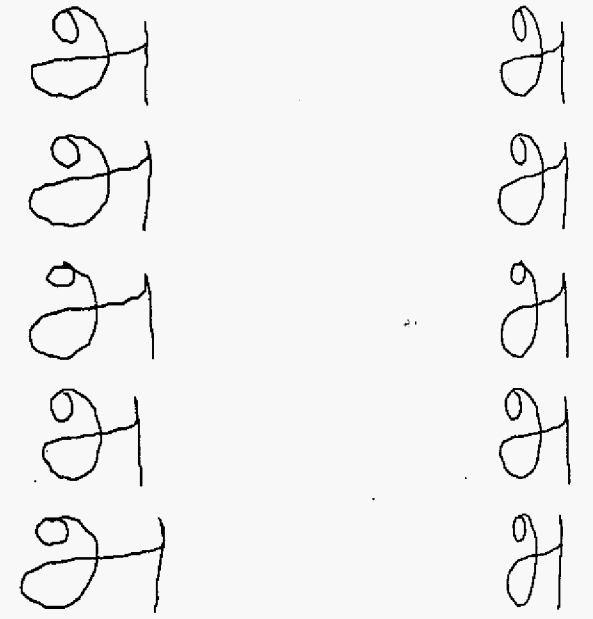
Population size	100
maximum number of generations	50
creation of initial random population	grow
Probability of mutation	0.01
Maximum depth for initial trees	6
Maximum depth for trees created by crossover	17
Selection	Fitness proportionate

10 samples of each class. This database is divided into three sets, each containing data from five writers. Set-I was used for evolving the GPs, while Set-II and Set-III are used for evaluating the results. A nearest neighbour classifier is used for the character recognition .

Table II gives various functions that evolved from different simulations with their corresponding classifier error rates for different datasets. The reduction in error rate as compared to the standard rescale-to-unit square method is obvious. Interestingly, all the GPs evolved are simple arithmetic expressions. Figure 3 shows results are optimized by preprocessing a character with the function (S2 + K2).

TABLE II
COMPARISON OF DIFFERENT EVOLVED RESCALING FUNCTIONS

Expression evolved	used for evolving GP	used in evaluating the GP	
	set-I	set -II	set -III
S	5.04%	4.35%	5.92%
S+Min	4.97%	4.43%	5.55%
(S2-Min)/S2	4.31%	3.52%	4.51%
S2 + K2	4.68%	3.11%	3.81%
rescaling to unit square	5.13%	8.41%	7.04%



original characters rescaled characters

Fig. 3. Rescaling using the evolved function (S2 + K2)

V. Summary

The results indicate that GP is capable of evolving rescaling functions that significantly improve the recognition accuracy of handwritten characters. A fundamental advantage of the technique is that the evolved GPs take into account the idiosyncrasies of the classification performed by the classifier on the given set without any explicit knowledge of its internals.

The space of functions explored is apparently sufficient for rescaling a velocity compensated resampled handwritten character. However the rescaling functions does not consider any information regarding the neighbouring points or the direction of the trace. Further explorations can include developing a complete pre-processing algorithm using GPs. This will include re-sampling (conversion of the variable dimension representation to a fixed length representation), global transformations like rotation and smoothing functions. The use of GP for evolving general transformations of characters using a variety of local and global operators is an interesting challenge.

References

[1] T. et. al. The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):787–808, 1990.

[2] J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[3] A. Teredesai, J. Park, and V. Govindaraju. Active handwritten character recognition using genetic programming. In *Proceedings of the 4th European Conference on Genetic Programming*, pages 371–380. Springer-Verlag, 2001.